Comparison of Software Architecture Styles using Quality Attributes

S. Angeline Julia, N. Snehalatha, Paul Rodrigues

Abstract:- ***Every software system has an architecture because every system can be shown to be composed of components and relations among them. The Software architecture of a program is the structure of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. Architecture styles plays a dominant role in solving complex systems. Here we analyzed these architecture styles using the quality attributes and the survey is given. From our analysis we conclude that software architecture which is flexible is very important in developing complex distributed applications.***

Key Words: ***Software Architecture, Styles, Quality Attributes.***

I. INTRODUCTION

Software Architecture involves the description of elements from which systems are built, interactions among those elements, patterns that guide their composition, and constraints on these patterns.

Generally, a particular system is defined in terms of a collection of components and interactions among those components. Components are clients and servers, databases, filters, and layers in a hierarchical system. Interactions such as client server protocols, database accessing protocols, asynchronous event multicast and piped streams.

Architectural style determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of constrains on how they can be combined.

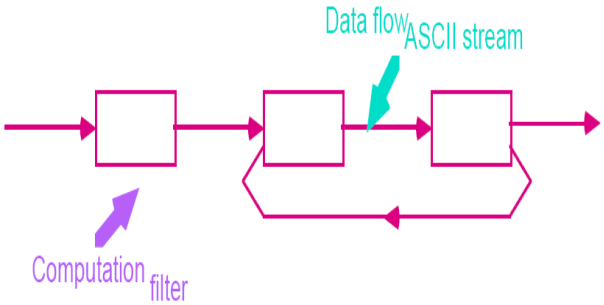
II. ARCHITECTURAL STYLES

Architectural styles can be divided into many types. They are dataflow systems, Call and return systems, Independent components and Network based systems. The Network based Systems can be further divided into Hierarchical styles and Replication styles. The explanations for all these types are given below.

Dataflow Systems

A. Pipes-and-Filters

Each component has a set of inputs and outputs. A component reads a stream of data on its input and produces a stream of data on its outputs. Input is transformed both locally and incrementally so that output begins before input is consumed (a parallel system). Components are called *filters*. Connectors serve as conduits for the information streams and are termed *pipes*.



Advantages

- Support reuse.
- Easy to maintain and enhance.

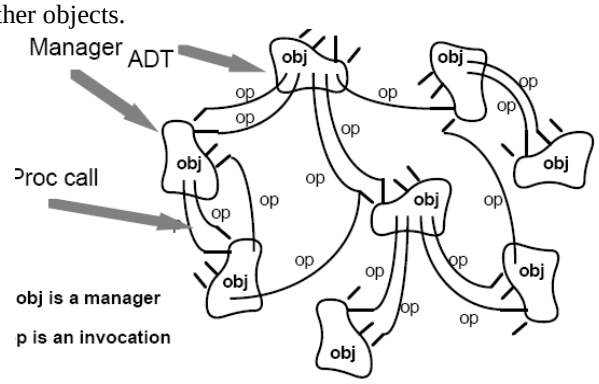
Disadvantage

- Poor for interactive applications

Call and Return Systems:

B. Data Abstraction and Object- Oriented Organization

Data and their associated operations are encapsulated into an abstract data type (object). The *components* of this style are the objects and connectors operate through procedure calls (methods). Two important characteristics of this style are: objects are responsible for maintaining the integrity of a resource and the representation of the object is hidden from other objects.



Advantage

- Hidden implementation details which allow the object to be changed without affecting its clients.

Disadvantage

- For an object to interact with another object it must know identity of that other object (unlike pipe-and-filter systems)
- Side-effects can occur if two objects use a common third object (if A use B and C uses B, then C's effect on B can cause unexpected side-effects on A).

Independent Components:

C. Event-Based, Implicit Invocation

Instead of invoking a procedure directly, a component announces (broadcasts) one or more events.

Manuscript revived on March 2013.

Mrs. S. Angeline Julia, Dept. of Software Engineering, SRM University, Chennai, Tamil Nadu, India.

Mrs. N. Snehalatha, Dept. of Software Engineering, SRM University, Chennai, Tamil Nadu, India.

Dr. Paul Rodrigues, Dean Research, Vellammal University, Chennai, TamilNadu, India.

Other components in the system can register interest in an event by associating a procedure with it. The system invokes all events which have registered with it. Event announcement ``implicitly" causes the invocation of procedures in other models. Implicit invocation systems are used in:

- programming environments to integrate tools
- user interfaces to separate data from representation

Advantage

- Invocation facilitates reuse
- Eases system evolution by allowing components to be replaced without affecting the interfaces of other components in the system.

Disadvantage

- The components relinquish control over the computation performed by the system. A component cannot assume that other components will respond to its requests and does not know what order events will be processed.
- In systems with a shared repository of data the performance and accuracy of the resource manager can become critical.

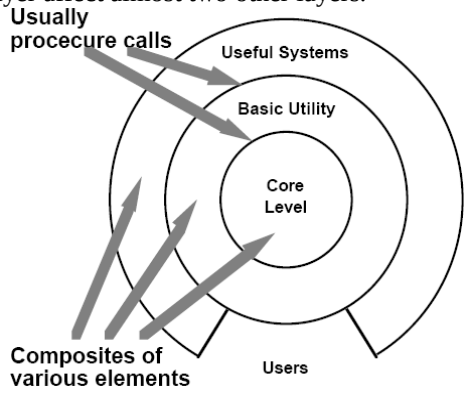
Call and Return Systems:

D. Layered Systems (LS)

A layered system is organized hierarchically with each layer providing service to the layer above it and serving as a client to the layer below. In some systems inner layers are hidden from all except the adjacent outer layer. Connectors are defined by the protocols that determine how layers will interact. Constraints include limiting interactions to adjacent layers. The best known example of this style appears in layered communication protocols OSI-ISO (Open Systems Interconnection - International Standards Organization) communication system. Lower levels describe hardware connections and higher levels describe application.

Advantage:

- Complex problems may be partitioned into a series of steps.
- Enhancement is supported since each layer interacts with almost the layers below and above, changes to the function of one layer affect almost two other layers.



- It improves evolvability and reusability.

Disadvantage:

- Difficulty in structuring some systems in a layered fashion.
- Add overhead and latency to the processing of data reducing user perceived performance.

Network Based System:

E. Client –Server (CS)

Each instance of the client software can send data requests to one or more connected *server*s. In turn, the servers can accept these requests, process them, and return the requested information to the client. A server machine is a high-performance host that is running one or more server programs which share its resources with clients. A client does not share any of its resources, but requests a server's content or service function.

Client examples are Web browsers, email clients and online chat. Server examples are web servers, ftp servers and database servers.

Advantages

- Greater ease of maintenance. For example, it is possible to replace, repair, upgrade, or even relocate a server while its clients remain both unaware and unaffected by that change.
- All data is stored on the servers, which generally have greater security controls than most clients. Clients with the appropriate permissions may access and change data.
- Since data storage is centralized, updates to that data are far easier to administer.
- It functions with multiple different clients of different capabilities.

Disadvantages

- As the number of simultaneous client requests to a given server increases, the server can become overloaded.
- If server failure occurs then client's requests cannot be fulfilled.

F. Layered Client Server (LCS)

Same as layered system but adds proxy and gateway to the client server style. Proxy acts as a shared server for one or more client components, taking requests and forwarding them with possible translation to the server components. A gateway component appears to be a normal server to the clients or proxies that request its services, but is in fact forwarding those requests with possible translation to its inner layer servers.

Advantage:

- Since this feature is added in multiple layers to add features like load balancing and security checking to the system.

G. Client Stateless Server (CSS)

Derives from client server (CS) with the additional constraint that no session state is allowed on the server component. Each request from client to server must contain all of the information necessary to understand the request and cannot take advantage of any stored context on the server. Session state is kept entirely on the client.

Advantage:

- Improve visibility. A monitoring system does not have to look beyond a single request datum in order to determine the full nature of the request.
- Improve reliability. It eases the task of recovering from partial failures.
- Improve scalability. Not having to store state between requests allows the server component to quickly free resources and further simplifies implementation.

H. Client cache stateless server (C$SS)

Derives from client stateless server and cache style. A cache act as a mediator between client and server in which the responses to prior requests can, if they are considered cacheable be reused in response to later requests that are equivalent and likely to result in a response identical to that in the cache if the request forwarded to the server.

Advantages:

- Eliminate partially or completely some interactions improving efficiency and user perceived performance.

I. Layered Client Cache Stateless Server (LC$SS)

Derived from both layered client server (LCS) and client cache stateless server (C$SS) through the addition of proxy and or gateway components.

J. Replicated Repository (RR)

Improve the accessibility of data and scalability of services by having more than one process provide the same service. These decentralized servers interact to provide clients the illusion that there is just one centralized service.

Advantage:

- Improve user perceived performance. Done by reducing the latency of normal requests and enabling disconnected operation in the face of primary server failure.
- More than one process provide the same service (scalability).
- Maintaining consistency is the primary concern.

K. Cache ($)

Replication of the result of an individual request such that it may be reused by later requests.

Advantage:

- Provide slightly less improvement than the replicated repository style in terms of user perceived performance. Since more requests will miss the cache and recently accessed data will be available for disconnected operation.
- Easy to implement. Does not require as much processing and storage.
- More efficient because data is transmitted only when it is requested.

III. QUALITY ATTRIBUTES

A. Maintainability

Maintainability/ modifiability are about the ease with which a change can be made to application architecture.

B. Reusability

Extent to which a program [or parts of a program] can be reused in other applications—related to the packaging and scope of the functions that the program performs.

C. Performance

Refers to the responsiveness of the system – the time required to stimuli or the number of events processed in some interval of time.

D. Simplicity

The primary means by which architectural styles induce simplicity is by applying the principle of separation of concerns to the allocation of functionality within components.

E. Scalability

Scalability refers to the ability of the architecture to support large numbers of components, or interactions among components, within an active configuration.

F. Portability

Effort required transferring the program from one hardware and/or software system environment to another.

G. Evolvability

Represents the degree to which a component implementation can be changed without negatively impacting other components.

H. User Perceived Performance

In user-perceived performance, the performance of an action is measured in terms of its impact on the user in front of an application rather than the rate at which the network moves information. The primary measures for user perceived performance are latency and completion time.

I. Visibility:

The ability of a component to monitor or mediate the interaction between two other components. Visibility can enable improved performance via shared caching of interactions, scalability through layered services, reliability through reflective monitoring, and security by allowing the interactions to be inspected by mediators.

J. Reliability:

The extent to which a program can be expected to perform its intended function with required precision.

K. Efficiency:

The amount of computing resources and code required by a program to perform its function.

IV. COMPARISION OF VARIOUS ARCHITECTURAL STYLES

Comparison between different architectural styles with respect to the quality attributes was done and the tabular format is given in the appendix 1 and appendix 2.

V. CONCLUSION

Software architecture is an essential level in process development of large and complicated system. Here architectural styles and the attributes are reviewed and the comparisons of these styles are given. The existing styles are not flexible enough to support many applications. Hence a new style which satisfies more quality attributes should be found to reach more success in software architecture styles.

REFERENCES

1. David Garlan and Mary Shaw,” Software Architecture perspectives on an emerging discipline” Prentice Hall of India private limited.
2. David Garlan and Mary Shaw,” An Introduction to Software Architecture”.
3. Len Bass, Paul Clements, Rick Kazman,” Software Architecture in practice” Addison-Wesley Longman, Inc.
4. Emad Majidi,Mahdieh Alemi, Hassan Rashidi, “Software Architecture: A Survey and Classification”, 2010 Second International Conference on Communication Software and Networks,2010 IEEE.

5. Francisca Losavio, Nicole Levy "Quality characteristics for Software Architecture", Journal of Object Technology, 2003.
6. Mikael Svahnberg, Claes Wohlin " A Comparative Study of Quantitative and Qualitative Views of Software Architectures" Proceedings EASE : Emprical Assessment and Evaluation in Software Engineering,Keele,UK,2003.
7. Roy Thomas Fielding "Architectural Styles and the Design of Network-based Software Architectures" 2000.
8. www.ieee.org
9. www.wikipedia.org/software architecture
10. www.bredemeyer.com/Architecture
11. http://msdn.microsoft.com/en-us/library/ee658117.aspx
12. http://coronet.iicm.tugraz.at/sa/s5/sa_styles.html
13. http://www.c-sharpcorner.com/uploadfile/kalisk/software-architecture-styles/

AUTHORS PROFILE

Mrs. S. Angeline Julia, is an Assistant professor in the Dept. of Software Engineering, SRM University, Chennai, Tamil Nadu, India. She has received her B.E in computer science engineering from Manonmaniam Sundaranar University, India & M.E in Embedded Systems Technologies from Anna University, India. She has total 6 years of Teaching experience. She is persuing Ph.D in Software Engineering.

Mrs. N. Snehalatha, is an Assistant professor in the Dept. of Software Engineering, SRM University, Chennai, Tamil Nadu, India. She has received her B.E in computer science engineering from Madras University, India & M.E in computer science engineering from Annamalai University, India. She has total 6 years of Teaching experience. She is persuing Ph.D in Wireless Network.

Dr.Paul Rodrigues, is Dean Research, Vellammal University, Chennai, India and CTO of WisdomTree Software Solutions, Chennai, India. He has received his B.Tech from Karnataka University, India & M.Tech from NIT-Allahabad, India and Ph.D from Pondicherry University, India. He has total 20 years of Teaching and Industry experience in Delivery Management, Software Engineering, Budget Management and Business Development. He has published more than 50 (refereed) papers in International Conferences/Journals which include Extreme Programming, Software Architecture, Databases and Object Oriented Analysis and Design. Also, he was an author of a chapter in the book "Recent Trends in Network Security & Applications" during July 2010 published by Springer Berlin Heidelberg publications, New York, USA. He is first in the world to apply Vastu to Software Architecture. He has worked in various domains that include Insurance, Retail, Digital Forensic, Content Management and Application Migrations. He is an active member of many professional-bodies like Identity Research Group, PMP, and CISSP.

Appendix 1:

Comparison of network based system styles:

Sl. No.	Quality Attributes	C/S	LS	LCS (C/S + LS)	CSS (CS)	RR	$	C$SS (CSS+$)	LC$SS (LCS+C$SS)
1	Simplicity	Data storage is centralized (+)		Do (+)	Do (+)		Does not require more processing and storage (+)	Do (++)	Do (+++)
2	Scalability	Support multiple diff. clients (+)	Interface to the adjacent layer (+)	Do (++)	Do Server won't store state between requests (++)	More than one process provide the same service (+)	Can be combined with CSS style (+)	Do(++)	Do (++++)
3	Evolvability	Easy to upgrade, repair & relocate (+)	Hide the inner layer from outer layer (+)	Do (++)	Do (+)			Do(+)	Do (+++)
4	Reusability		Do (+)	Do (+)					Do (+)
5	User Perceived Performance		Reduce coupling across multiple layers (_)	Do (_)		Reduce latency and enable disconnected operation(+)	Recently accessed data will be available for disconnected system (+)	Eliminate some interactions(+) do(+)	Do (±+)

6	Visibility				All data needed are send from client - server(+)			Do (+)	Do (+)
7	Reliability				Easy recovery from failure (+)	Maintainc onsistency (+)		do (+)	Do (+)
8	Efficiency						Data transmitte d when required (+)	Eliminate some interaction s(+) do (+)	Do (++)

Appendix 2:

Comparison of Architectural styles based on the attributes:

Styles	Simplicity	Maintainability	Reuse	Performance	Scalability	Portability
Pipe & Filter	Easy to use (+)	Replacement of filters easy (+)	Achieve different effects by recombination (+)	No interaction between filters (-)		Not portable Identity not known (-)
Object Oriented Organization		Encapsulation of data is done (+)	Access of data is done through methods (polymorphism) (+)			
Layered System		Changeability in a layer affect adjacent two layers only (+)	Layers have no dependencies on higher layers (+)	Some functions require close coupling (±)	Can be combined with other styles (+)	
Event based Implicit invocation	Sometimes unpredictable & hard to control (-)	Replaced without affecting other components (+)	Components can be registered in a system for any event (+)	Components have no control over the computation (-)	Side effects occur if two components use same third component (-)	Components can be combined with any events (+)